ACM321 Java Lab-7 Notes

Inheritance (is-a relationship)

Java program are written as collection of **classes**, which serve as templates for individual **objects**.

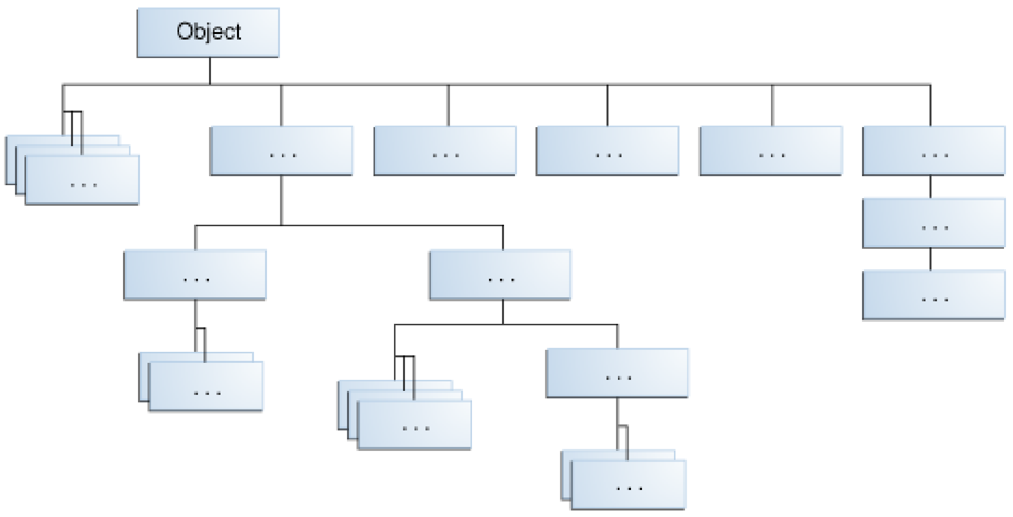
Each object is an **instance** of a particular **class**.

Classes in Java form **hierarchies**, except for the class named **Object** that stands at the top of the hierarchy.

Every class in Java is a **subclass** of some other class, which is called its superclass. A class can have many subclasses, but each class has only one superclass.

The **Object** class, defined in the java.lang package, defines and implements behavior common to all classes—including the ones that you write. In the Java platform, many classes derive directly from **Object**, other classes derive from some of those classes, and so on, forming a hierarchy of classes.

> Other definition from docs.oracle.com
> → Classes can be derived from classes that are derived from classes that are derived from classes, and so on, and ultimately derived from the topmost class, Object. Such a class is said to be descended from all the classes in the inheritance chain stretching back to Object.



All Classes in the Java Platform are Descendants of Object

A class can have many subclasses, but each class has only one superclass.

When you define a **new class** in Java, that class automatically **inherits** the behavior of its **superclass**.

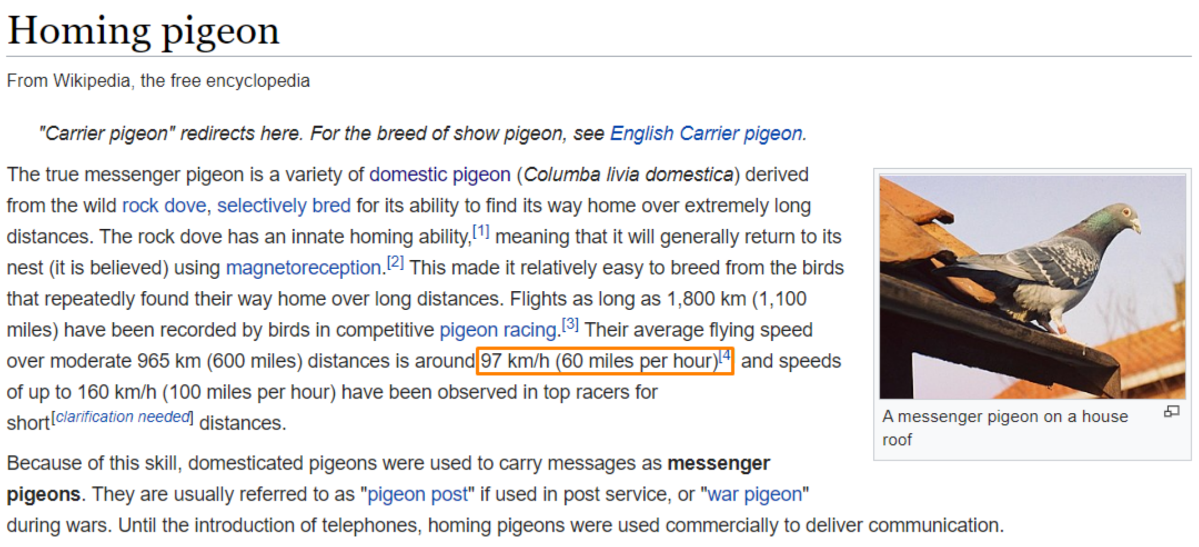
It can also be defined as; → The class from which the **subclass is derived** is called a **superclass** (also a base class or a parent class).

An Example of Inheritance

```
//Animal.java
package lab7;

public class Animal {

    // the Animal class has three fields
    private boolean vegetarian;
    private int number_of_legs;
    private String habitat;

    // the Aniamal class has one constructor
    public Animal(boolean vegetarian, int number_of_legs, String habitat) {
        this.vegetarian = vegetarian;
        this.number_of_legs = number_of_legs;
        this.habitat = habitat;
    }

    // the Animal class has three set methods
    public void setVegeratian(boolean vegetarian) {
        this.vegetarian = vegetarian;
    }

    public void setNumberofLegs(int number_of_legs) {
        this.number_of_legs = number_of_legs;
    }

    public void setHabitat(String habitat) {
        this.habitat = habitat;
    }

    // the Animal class has three get methods
    public boolean getVegeratian() {
        return this.vegetarian;
    }

    public int getNumberofLegs() {
        return this.number_of_legs;
    }

    public String getHabitat() {
        return this.habitat;
    }

}
```

```
//Bird.java
package lab7;

public class Bird extends Animal {

    // the Bird subclass adds one field
    private String color;
    private int speed;

    // the Birds subclass has one constructor
    public Bird(boolean vegetarian,
        int number_of_legs,
                String habitat,
                String color,
                int speed) {
        super(vegetarian, number_of_legs, habitat);
        this.color = color;
    this.speed = speed;
    }

    // the Bird subclass adds two set methods
    public void setColor(String color) {
        this.color = color;
    }

    public void setSpeed(int speed) {
        this.speed = speed;
    }

    // the Bird subclass adds two get methods
    public String getColor() {
        return this.color;
    }

    public int getSpeed() {
        return this.speed;
    }
}
```

Lets create a **piegon** object from **Bird** class which is derived from **Animal** superclass.



Homing pigeon

From Wikipedia, the free encyclopedia

"Carrier pigeon" redirects here. For the breed of show pigeon, see English Carrier pigeon.

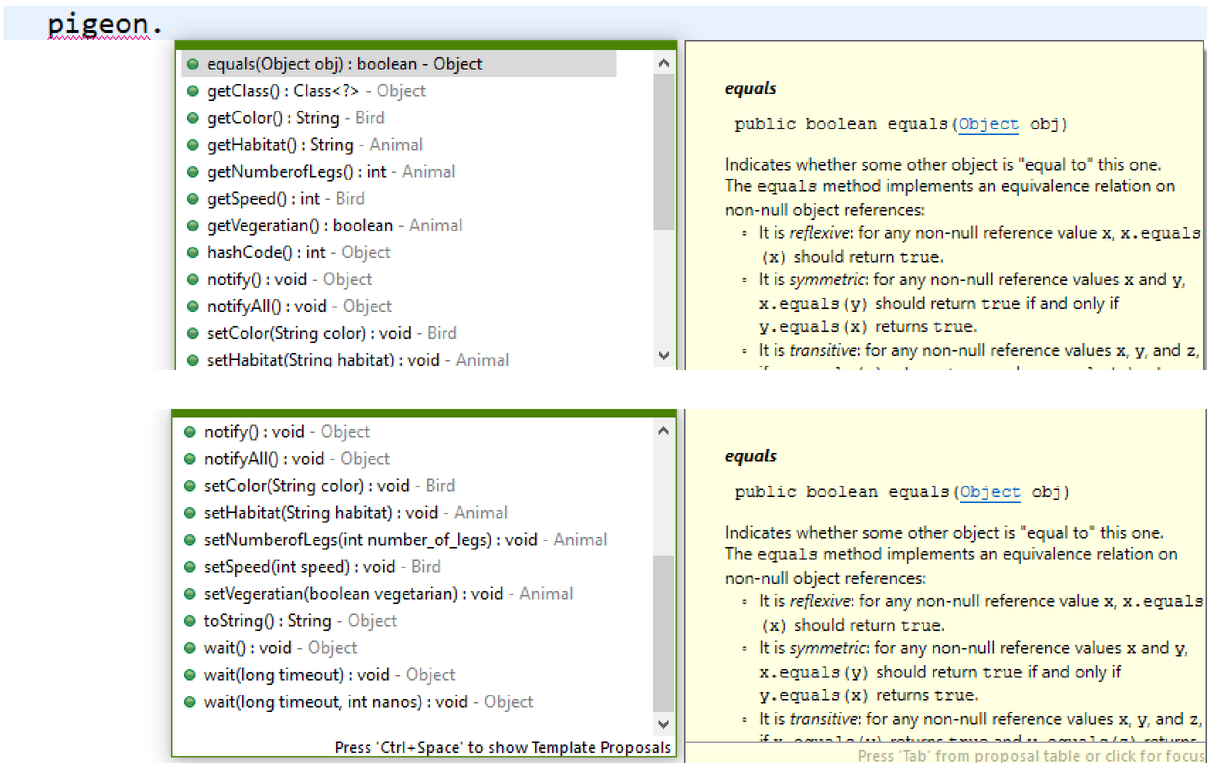
The true messenger pigeon is a variety of domestic pigeon (*Columba livia domestica*) derived from the wild rock dove, selectively bred for its ability to find its way home over extremely long distances. The rock dove has an innate homing ability,[1] meaning that it will generally return to its nest (it is believed) using magnetoreception.[2] This made it relatively easy to breed from the birds that repeatedly found their way home over long distances. Flights as long as 1,800 km (1,100 miles) have been recorded by birds in competitive pigeon racing.[3] Their average flying speed over moderate 965 km (600 miles) distances is around 97 km/h (60 miles per hour)[4] and speeds of up to 160 km/h (100 miles per hour) have been observed in top racers for short[clarification needed] distances.

A messenger pigeon on a house roof

Because of this skill, domesticated pigeons were used to carry messages as **messenger pigeons**. They are usually referred to as "pigeon post" if used in post service, or "war pigeon" during wars. Until the introduction of telephones, homing pigeons were used commercially to deliver communication.

```
//InheritenceDemo.java
package lab7;

public class InheritenceDemo {

  public static void main(String[] args) {
    //pigeon means guvercin
    Bird pigeon = new Bird(true, 2, "Air", "Grey", 97);
    Bird feral_pigeon = new Bird(true, 2, "Air", "Yellow", 40);

    String printable_object_output = pigeon.toString();
    String printable_object_output2 = feral_pigeon.toString();

    System.out.println(printable_object_output);
    System.out.println(printable_object_output2);

    System.out.println("Renk: " + pigeon.getColor() + " Habitat: " + pigeon.getHabitat());
    System.out.println("Renk: " + feral_pigeon.getColor() + " Habitat: " + feral_pigeon.getHabitat());
  }

}
```

Some of the methods on **pigeon**, we didn't declare for example **equals(), getClass(), notify(), ... , toString()**



```
pigeon.
```

- equals(Object obj) : boolean - Object
- getClass() : Class<?> - Object
- getColor() : String - Bird
- getHabitat() : String - Animal
- getNumberofLegs() : int - Animal
- getSpeed() : int - Bird
- getVegeratian() : boolean - Animal
- hashCode() : int - Object
- notify() : void - Object
- notifyAll() : void - Object
- setColor(String color) : void - Bird
- setHabitat(String habitat) : void - Animal

equals

`public boolean equals(Object obj)`

Indicates whether some other object is "equal to" this one.
The `equals` method implements an equivalence relation on non-null object references:
- It is *reflexive*: for any non-null reference value `x`, `x.equals(x)` should return `true`.
- It is *symmetric*: for any non-null reference values `x` and `y`, `x.equals(y)` should return `true` if and only if `y.equals(x)` returns `true`.
- It is *transitive*: for any non-null reference values `x`, `y`, and `z`,

- notify() : void - Object
- notifyAll() : void - Object
- setColor(String color) : void - Bird
- setHabitat(String habitat) : void - Animal
- setNumberofLegs(int number_of_legs) : void - Animal
- setSpeed(int speed) : void - Bird
- setVegeratian(boolean vegetarian) : void - Animal
- toString() : String - Object
- wait() : void - Object
- wait(long timeout) : void - Object
- wait(long timeout, int nanos) : void - Object

Press 'Ctrl+Space' to show Template Proposals

equals

`public boolean equals(Object obj)`

Indicates whether some other object is "equal to" this one.
The `equals` method implements an equivalence relation on non-null object references:
- It is *reflexive*: for any non-null reference value `x`, `x.equals(x)` should return `true`.
- It is *symmetric*: for any non-null reference values `x` and `y`, `x.equals(y)` should return `true` if and only if `y.equals(x)` returns `true`.
- It is *transitive*: for any non-null reference values `x`, `y`, and `z`,

Press 'Tab' from proposal table or click for focus

pigeon actually is an instance of Bird class. It has methods of Animal class because the Bird class inherited from Animal class. In other words, Animal class is the superclass of Bird class.

In addition, the pigeon couldn't get the properties of the Animal class. The reason behind that they are private and we only can access them via public methods.

Also see section below, **Private Members in a Superclass**.

What You Can Do in a Subclass

- The inherited fields can be used directly, just like any other fields.
- You can declare new fields in the subclass that are not in the superclass.
- The inherited methods can be used directly as they are.
- You can write a new instance method in the subclass that has the same signature as the one in the superclass, thus overriding it.
- You can declare new methods in the subclass that are not in the superclass.

Private Members in a Superclass

A subclass **does not inherit** the **private members of its parent class.**

However, if the superclass has **public** or **protected methods** for **accessing its private fields**, these can also be used by the subclass.

TR: Burada bizim yaptığımız da budur, private olarak tanımlanmış öznitelikler kalıtım yolu ile taşınmaz fakat eğer o özniteliklere public erişebilen (accessors or mutators) yani get veya set metodları varsa, kalıtım yolu ile taşınırlar.
Zaten Bird class ının Animal class ına erişemediği özellikleri (private) alması zaten mümkün değil. Onlara birer public methodlarla erişebildiğini gördüğü için bu özellikleri alabiliyor.

Overriding Methods in Java

Lets add a method talk to **Animal** class.

```
//Animal.java
package lab7;

public class Animal {

    // the Animal class has three fields
    private boolean vegetarian;
    private int number_of_legs;
    private String habitat;

    // the Aniamal class has one constructor
    public Animal(boolean vegetarian, int number_of_legs, String habitat) {
        this.vegetarian = vegetarian;
        this.number_of_legs = number_of_legs;
        this.habitat = habitat;
    }

    //..... same codes here as above, add next lines below
    //new method talk, it returns information about itself as a string.
    //this method will be overrided.
    public void talk() {
      System.out.println(this.toString());
    }

}
```

Lets modify Bird class with the same named method to override it for Bird.

```
//Bird.java
package lab7;

public class Bird extends Animal {

    // the Bird subclass adds one field
    private String color;
    private int speed;

    // the Birds subclass has one constructor
    public Bird(boolean vegetarian,
        int number_of_legs,
                String habitat,
                String color,
                int speed) {
        super(vegetarian, number_of_legs, habitat);
        this.color = color;
    this.speed = speed;
    }

    //..... same codes here as above, add next lines below
    //Overriding method talk(), it returns two word to talk.

    @Override
    public void talk() {
      System.out.println("Cik cik");
    }

}
```

```
//InheritanceDemo.java

package lab7;

public class InheritenceDemo {

  public static void main(String[] args) {

    Animal animal_obj = new Animal(false, 2, "Land");

    //pigeon means guvercin
    Bird pigeon = new Bird(true, 2, "Air", "Grey", 97);

    animal_obj.talk();
    //Output: lab7.Animal@7852e922

    pigeon.talk();
    //Output: Cik cik

  }

}
```